Problem Statement

1. Use timer to simulate data generated by external sensor. Setup the timer to trigger every second and generates random number (0 to 5) of random bytes and adds this to a globally accessible data structure.

2. Separately wake up periodically (every 10s), checks if 50 bytes are stored in the globally accessible data structure and prints only the latest 50 bytes (in hex value) and deletes the printed bytes from the data structure.

3. For example, 1st second 4 bytes are added, 2nd second 3 bytes are added and 10th second there are 39 bytes are in the buffer. Thus at 10th second data is not printed. At 20th second, if there are more than 50 bytes in the buffer, the main thread only prints the latest 50 bytes and deletes them.

Understanding the Problem Statement

From the problem statement, what I understood is that, generate a random number every second and use it to generate that many random bytes, appending them to a list.Every 10 seconds, if the list has at least 50 bytes, print the latest 50 in hex and delete them, then stop the program.

Code

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <unistd.h>

#define MAX_SIZE 1000

unsigned char data[MAX_SIZE];
int count = 0;

void addData() {
```

```
    int n = rand() % 6;
    for (int i = 0; i < n && count < MAX_SIZE; i++) {
        data[count++] = rand() % 256;
    }
}

void printLast50() {
    printf("Last 50 bytes in hex:\n");
    for (int i = count - 50; i < count; i++) {
        printf("%02X ", data[i]);
    }
    printf("\n");

    // Keep the older bytes (from index 0 to count-51)
    count -= 50;
}

int main() {
    srand(time(0));

    for (int sec = 1; sec <= 1000; sec++) {
        addData();
        sleep(1);

        if (sec % 10 == 0 && count >= 50) {
            printLast50();
            printf("End program.\n");
            break;
        }
    }

    return 0;
}
```

Code Explanation

- #include <stdio.h> – Used for standard input and output functions like printf.
- #include <stdlib.h> – Provides functions like rand() for generating random numbers.
- #include <time.h> – Used to seed the random number generator with srand(time(0)).
- #include <unistd.h> – Allows use of sleep() function to pause execution for 1 second.
- #define MAX_SIZE 1000 – Sets the maximum number of bytes the array (acting as buffer) can hold.
- unsigned char data[MAX_SIZE]; – A byte array to store all the randomly generated data.
- int count = 0; – Keeps track of the current number of bytes added to the buffer.
- This function generates a random number between 0 and 5 using rand() % 6 and stores it in n.
- Then a for loop runs n times to generate n random bytes using rand() % 256, which gives a value between 0 and 255.
- Each byte is added to the data[] array, and the count is incremented accordingly.
- The condition count < MAX_SIZE ensures we don't overflow the array.
- This function generates a random number between 0 and 5 using rand() % 6 and stores it in n.
- Then a for loop runs n times to generate n random bytes using rand() % 256, which gives a value between 0 and 255.
- Each byte is added to the data[] array, and the count is incremented accordingly.
- The condition count < MAX_SIZE ensures we don't overflow the array.
- This function is called only if the total number of bytes in the array (count) is 50 or more.

- It uses a loop to print the last 50 bytes from the array in hexadecimal format using printf("%02X ", data[i]).
- %02X means the byte will be printed as a 2-digit hexadecimal number, padded with zero if needed.
- After printing, count is reduced by 50 to simulate the removal of those 50 bytes from the end of the buffer.
- srand(time(0)); initializes the random number generator to produce different outputs on each run.
- A loop runs from sec = 1 to 1000, representing a simulated time in seconds.
- Inside the loop:
- addData() is called every second to add random bytes to the buffer.
- sleep(1); pauses execution for 1 second to simulate real-time behavior.
- Every 10 seconds (when sec % 10 == 0), it checks if there are at least 50 bytes in the buffer.
- If true, it calls printLast50() and prints “End program.” before breaking out of the loop.
- The program terminates immediately after printing the latest 50 bytes and does not run further.
- This satisfies the requirement to stop after the first valid 10-second check that meets the byte condition.